Learning Sampling Distributions for Efficient Object Detection

Yanwei Pang, *Senior Member, IEEE*, Jiale Cao, and Xuelong Li, *Fellow, IEEE*

***Abstract*—Object detection is an important task in computer vision and learning systems. Multistage particle windows (MPW), proposed by Gualdi *et al*., is an algorithm of fast and accurate object detection. By sampling particle windows from a proposal distribution (PD), MPW avoids exhaustively scanning the image. Despite its success, it is unknown how to determine the number of stages and the number of particle windows in each stage. Moreover, it has to generate too many particle windows in the initialization step and it redraws unnecessary too many particle windows around object-like regions. In this paper, we attempt to solve the problems of MPW. An important fact we used is that there is a large probability for a randomly generated particle window not to contain the object because the object is a sparse event relevant to the huge number of candidate windows. Therefore, we design the proposal distribution so as to efficiently reject the huge number of non-object windows. Specifically, we propose the concepts of rejection, acceptance, and ambiguity windows and regions. This contrasts to MPW which utilizes only on region of support. The PD of MPW is acceptance-oriented whereas the PD of our method (called iPW) is rejection-oriented. Experimental results on human and face detection demonstrate the efficiency and effectiveness of the iPW algorithm. The source code is publicly accessible.**

***Index Terms*—Object detection, particle windows, random sampling, feature extraction.**

I. Introduction

OBJECT detection is a key component of many computer vision systems [26], [45]. Generally, object detection consists of two steps: feature extraction and classification [44], [35], [3], [13]. In this paper, we divide the task of object detection into three steps: window generation, feature extraction, and classification. Window generation outputs windows determined by shape, location and size. Features are extracted from the windows and then are classified by a classifier. Suppose N windows are generated and the time spent in generating the windows is t_w. Let t_f be the time of extracting a feature vector from a window (i.e., subimage) and t_c be the time of classifying the feature vector as either positive or negative class. Then the computation time t of an object detection algorithm can be expressed as:

$$t = t_w + N \times t_f + N \times t_c. \quad (1)$$

Y. Pang and J. Cao are with the School of Electronic Information Engineering, Tianjin University, Tianjin 300072, China. e-mails: {pyw,connor}@tju.edu.cn

X. Li is with the Center for OPTical IMagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, Shaanxi, P. R. China. e-mail: xuelong_li@opt.ac.cn.

Usually, t_w is very small and can be neglected. Obviously, it is desirable if N is as small as possible on the condition that the detection rate and false positive rate are acceptable, which is the goal of our algorithm.

A dominant manner of window generation is sliding window (i.e., SW) based scanning. Given the pixel stride and scale factor, windows are determinately generated from top to bottom, left to right, and small to large. This deterministic manner requires a very large number of windows in order to detect objects with high detection rate.

Windows can also be generated in a stochastic (random) manner. The stochastic manner can also be categorized into active sampling [2]. Recently, Gualdi *et al*. [15] proposed to generate the windows (called particle windows) by sampling from a probability density function which is called proposal distribution (PD). This algorithm is called multistage particle window with abbreviation MPW. The initial PD is uniform distribution, meaning that each candidate window has the same chance to contain the object. If some windows of the N_1 sampled particle windows (called particle windows in [15]) have large classifier responses, then the PD is updated by enhancing the positions nearby these windows. Consequently, when sampling from the updated PD, more windows will be generated near the previous particle windows. Therefore, a smaller number $N_i (N_i < N_1)$ of particle windows is needed to be drawn from the updated PD, which is why the MPW method can use smaller number of windows to get the same detection rate and false positive rate as the SW method.

Despite the great success of MPW, there is still room to improve it. Due to the non-negativity of the weights and density, almost all particle windows are sampled from the regions neighboring to the particle windows obtained in the previous stages. Therefore, if the number of initial particle windows is not large enough to contain true positives, then there is a very large probability that MPW will not sample the positive windows any more. In addition, MPW generates unnecessary too many particle windows around the object and object-like regions. Considering that classification of these regions is more time-consuming than obvious non-object regions in the algorithm of cascade AdaBoost, so generating too many particle windows around the object and object-like regions greatly limits its efficiency. Importantly, it is unknown to use how many particle windows in each stage.

In this paper, we propose to improve MPW with the aim of sampling a smaller number of particle windows without any loss in detection rate and false positive rate. In addition,

we solve the problem of determining the number of particle windows in each stage. We call it iPW. Compared to MPW, iPW has the following characteristics and advantages.

1) iPW does not need to generate a large number of particle windows at the initial iteration (stage), which greatly reduces the detection time. Even if the initial particle windows do not contain any object, iPW can detect the objects at next stages.
2) MPW draws unnecessary too many particle windows around the positive windows whereas iPW avoids generating the redundant particle windows by using the information of both rejected and accepted particle windows.
3) To use MPW, one has to empirically set the number of particle windows in each stage whereas this is not a problem because iPW generates a single particle window in each iteration (stage).
4) iPW fully makes use of the information of rejected negative particle windows while MPW almost completely depends on the accepted positive particle windows. Rejected negative particle windows are used to directly suppress the PD around these windows and at the same time indirectly enhance the PD beyond the windows. In this sense, iPW is rejection-oriented while MPW is acceptance-oriented.
5) In MPW, the uniform distribution is used in the initialization stage and plays an unimportant role in the later stages so that it can be omitted. In iPW, a dented uniform distribution is used to play an important role for sampling useful particle windows by rejecting impossible regions.
6) iPW utilizes dented Gaussian distribution while MPW utilizes full Gaussian distribution for sampling particle windows. By using dented Gaussian distribution, iPW avoids drawing many unnecessary particle windows around the object and object-like regions.
7) To obtain the same detection accuracy, the total number of particle windows in iPW is much smaller than that in MPW.

The remainder of the paper is organized as follows: In Section 2, related work is discussed. Section 3 reviews the MPW algorithm. The proposed iPW algorithm is described in Section 4. Experimental results are given in Section 5 before summarizing and concluding in Section 6.

II. Related Work

According to (1), the computation time of an object detection algorithm is determined by window generation, feature extraction, classification, and the number of windows. Accordingly, we can categorize existing efficient object detection algorithms into feature-reduced, classification-reduced, and window-reduced types. In addition, combination of the different types of algorithm should also be considered.

Cascade AdaBoost plus Haar-like features can be viewed as classical combination of feature-reduced and classification-reduced method [41], [30]. In contrast, neural network based face detection [34] is time-consuming in feature extraction and classification though it has comparable detection accuracy. Similarly, Histogram of Oriented Gradients (HOG) plus Support Vector Machine (SVM) [12] can be improved in efficiency by using integral image and cascade structure. The trilinear interpolation of HOG can also be approximated by decomposing gradients into different angle planes followed by simple smoothing [28]. There are many efficient object detection algorithms using the technique of integral image for extracting simple but rich features. One important type of features for human detection is integral channel features [4], [9].

Designing optimal cascade structure is also an important topic for increasing the speed of object detection. Recent methods include *crosstalk cascade* [11], CoBE [5], LAC-Boost [36], [43], sparse decision DAGS (*directed acyclic graphs*) [7], etc.

In addition to the integral-image based algorithm, coarse-to-fine feature hierarchy [10], [46] and template matching with binary representation [16] are also feature-reduced methods. The coarse-to-fine feature hierarchy is able to reject the majority of negative windows by the lower resolution features and process a small number of windows by higher resolution features [46]. By deep analysis of statistic of multiscale features, Dollár *et al*. [10] developed an efficient scheme for computing feature pyramids. Liu *et al*. [24] developed a probability-based pedestrian mask which can be used as pre-filter to filter out many non-pedestrian regions. Template matching with binary representation for gradient information is a promising method for detecting textureless objects in real time [16]. Owing to its elegant feature representation and the architecture of modern computers, template matching with binary representation for gradient information can use thousands of arbitrarily sized and shaped templates for object detection in very fast speed [16]. By setting proper sliding stride, features can be reused to avoid computing the features in a window overlapping with its neighbors [32]. The information of spatial overlap can also be used for image matching and recognition[1]. Deep learning with rich features hierarchy is also a promising direction [14].

Classifier-reduced method arrives at high efficiency by designing efficient classifier. In addition to cascade AdaBoost which uses a few of classifier to reject a lot of windows, one can design efficient linear and nonlinear SVM to classification. Vedaldi and Zisserman proposed to use explicit, instead of implicit, feature maps to approximate nonlinear SVM [40]. Mak and Kung developed a low-power SVM classifier where the scoring function of polynomial SVMs can be written in a matrix-vector-multiplication form so that the resulting complexity becomes independent of the number of support vectors [25], [19]. Linear SVM classifies a feature vector by computing the inner product between the sum of weighted support vectors and the feature vector. To reduce the computation complexity of the inner-product based classification, Pang *et al*. proposed a sparse inner-product algorithm [33]. The idea is that neighboring sub-images are also neighboring to each other in the feature

space and have similar classifier responses. Pang *et al*. also developed a distributed strategy for computing the classifier response[26].

Window-reduced method is a promising direction developed in recent five years. This kind of methods aim at reducing the number of windows where feature extraction and classification have to be conducted. When an image is represented by a small number of keypoints and their descriptors (i.e., visual words), branch&bound (a.k.a., efficient subwindow wearch (ESS)) is very efficient because it hierarchically splits the parameter spaces into disjoint spaces and uses quality functions to reject large parts of the parameter space [20]. Branch&bound can also be used in implicit shape model which adopts hough-style voting [22], [23]. Branch&rank generalizes the idea of branch&bound by learning a ranking function that prioritises hypothesis sets that do contain an object over those that do not [21]. Acting testing is also a promising method for rapid object detection [2], [37]. But these visual-word based methods are not suitable for detecting textureless objects because it is not reliable to detect keypoints from the objects.

As a signal can be reconstructed from irregularly sampled points [27], an object can be detected by random sampling a fraction of all the possible locations and scales. Multi-stage particle window (MPW) is a window-reduced object detection method [15] using random sampling. Branch&rank and branch&bound are based on keypoint detection whereas MPW extracts Haar-like features, HOG, or other features as the same manner of sliding window based object detection method. Therefore, MPW is expected to be suitable for detecting both texture-rich and texture-less objects. Sliding window based method investigates all the windows overlappingly and uniformly spaced in spatial and scale domains. In contrast, MPW only checks the windows generated from an updating proposal distribution. As iteration proceeds, the main peaks of the distribution evolve towards the objects. The open problem in MPW is how many windows should be generated in each iteration (stage). Existing MPW uses empirical numbers which is hard to result in optimal solution. In this paper, we propose a novel particle window based object detection method that is more efficient than MPW and avoids choosing particle numbers in multiple stages.

It is noted that the technique of Detection Proposals (DP) [48] (sometimes called Objectness [8] or Selective Search [39]) also generates a number of windows by sampling from all the possible windows. But the number of generated windows has to be large (e.g., $10\hat{3}$ or $10\hat{4}$) enough if acceptable detection quality is required. Moreover, the time spend on generating detection proposals is not satisfying (see Table 2 of [17]). Nevertheless, the DP methods have been successfully employed in deep leaning based detection [38].

III. Multi-stage Particle Windows

The algorithm of multi-stage particle window (MPW) [15] is the basis of our method.

A. Algorithm

MPW investigates a fraction of all candidate windows in an image by sampling from a proposal distribution. Each particle window represents a window $\mathbf{w} = (x, y, s)^T$ where x, y and s are the horizontal position, the vertical position and the size of the window, respectively. The window can also be expressed as $\mathbf{w}(x, y, s)$. Once a window $\mathbf{w}$ is generated, the feature vector is then extracted and classified. Let $f(\mathbf{w})$ be the classifier response. The main issue of MPW is how to design the proposal distribution.

At the beginning of the MPW algorithm, no prior knowledge is known about the preference on the candidate windows. So the proposal distribution $q_0(\mathbf{w}) = q_0(x, y, s)$ is modeled as a uniform distribution $u(\mathbf{w}) = u(x, y, s) = 1/N$, where N is number of all possible windows in the image.

In the first iteration, N_1 particle windows are drawn from the uniform proposal distribution $q_0(\mathbf{w}) = u(\mathbf{w})$ (see Fig. 1(a)). The classifier response $f(\mathbf{w}_i)$ is normalized by

$$f(\mathbf{w}_i) \leftarrow \frac{f(\mathbf{w}_i)}{\sum_{j=1}^{N_1} f(\mathbf{w}_j)}, \tag{2}$$

so that $\sum_{i=1}^{N_1} f(\mathbf{w}_i) = 1$, and $0 \leq f(\mathbf{w}_i) \leq 1$. Then the proposal distribution is updated according to the classifier responses $f(\mathbf{w})$ of the N_1 particle windows:

$$q_1(\mathbf{w}) = (1 - \alpha_1) q_0(\mathbf{w}) + \alpha_1 \sum_{j=1}^{N_1} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma). \tag{3}$$

In (3), $G(\mathbf{w}_j, \Sigma)$ is a Gaussian distribution where the mean is centered at $\mathbf{w}_j$ and Σ is the covariance of Gaussian distribution. The weight α_1 balances the previous proposal distribution q_0 and the mixture of Gaussian distributions. Gualdi *et al*. [15] found that $\alpha_1 = 1$ is almost the best choice, meaning the unimportance of previous proposal distribution. In Section 3.2 we will explain why $\alpha_1 = 1$ is a reasonable choice.

The sum term in (3) is called measurement density function $p_1(\mathbf{w})$ [15] :

$$p_1(\mathbf{w}) = \sum_{j=1}^{N_1} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma). \tag{4}$$

If $\alpha_1 = 1$, then the proposal distribution $q_1(\mathbf{w})$ is identical to the measurement density function $p_1(\mathbf{w})$.

In the second iteration, N_2 particle windows are drawn from $q_1(\mathbf{w})$. Usually, N_2 is smaller than N_1. Because the classifier response $f(\mathbf{w})$ is large in the regions nearby the positives, most of the N_2 particle windows lie around the positives (see Fig. 1(b)).

The iteration continues with the new proposal distribution in stage i:

$$q_i(\mathbf{w}) = (1 - \alpha_i) q_{i-1}(\mathbf{w}) + \alpha_i \sum_{j=1}^{N_i} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma_i). \tag{5}$$

If $\alpha_i = 1$, the proposal distribution becomes:

$$q_i(\mathbf{w}) = p_i(\mathbf{w}) = g_i(\mathbf{w}) = \sum_{j=1}^{N_i} f(\mathbf{w}_j) G(\mathbf{w}_j, \Sigma_i), \tag{6}$$

which is in fact the employed proposal distribution in the experiments in [15].

Algorithm 1 shows the procedure of MPW.

(a) (b)

Fig. 1. The process of MPW. (a) Stage 1 samples initial particle windows by uniform distribution. (b) Stage 2 generates particle windows around pedestrian.

Algorithm 1 The algorithm of MPW.

Input:

Stage number S;

The number N_i of particle windows in stage i, $i = 1, \ldots, S$;

The number N of all candidate windows.

Output:

The set $\mathbf{W}_P$ of positive particle windows.

Initialization
Empty the set of positive particle windows: $\mathbf{W}_P \leftarrow \Phi$.
Initialize the proposal distribution $g(\mathbf{w})$ by uniform distribution $u(\mathbf{w}) = 1/N$, i.e., $g(\mathbf{w}) \leftarrow 1/N$.
for $s = 1$ **to** S **do**
  Sample N_i particle windows from $g(\mathbf{w})$.
  Put the N_i particle windows into $\mathbf{W}$, i.e., $\mathbf{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_{N_i}\}$.
  If $f(\mathbf{w}_j) = 1$, j=1, $\ldots$, N_i, then $\mathbf{W}_P = \mathbf{W}_P \cup \mathbf{w}_j$.
  Update $g(\mathbf{w})$ using the $\mathbf{W}$, and empty $\mathbf{W}$.
end for
return $\mathbf{W}_P$.

B. Why $\alpha_i = 1$

In this section, we explain why $\alpha_i = 1$ is a reasonable choice. To the best of our knowledge, we are the first to explain why $\alpha_i = 1$.

For the sake of simplicity, we assume that the scale is fixed and there is only one object in the image. For an $h \times w$ image, the number of candidate windows is $M = h \times w$. Let the size of support region be $m \ll M$. In the initialization step, N_1 particle windows are drawn from the uniform distribution $q_0(\mathbf{w}) = u(\mathbf{w})$. Then the probability p that the N_1 particle windows contain the object is $p = 1-(1-m/M)^{N_1}$. Suppose that $M = 640 \times 480$, $m = 50$, and $N_1 = 1000$, then $p = 0.15$. Obviously, if N_1 is small, it is a small probability that particle windows contain the object.

In the later stage, MPW generates a smaller number of particle windows, where $(1 - \alpha_1)$ fraction is from the uniform distribution. The probability that the $(1 - \alpha_1)$ fraction of particle windows contains the object is much smaller. So α_i is usually set 1.

TABLE I
REPRESENTATIVE VALUES OF N_i

Stage number i	1	2	3	4	5
N_i	2000	1288	829	534	349

C. Merits and Limitations

Compared to SW, MPW is able to obtain similar accuracy at lower computational load [15]. However, it is not clear that how to optimally select the number m of stages and the number N_i of particle windows in each stage. Guadldi *et al*. gives an empirical rule for parameter selection:

$$N_i = N_1 \times e^{-\gamma(i-1)}, i = 1, ..., m, \tag{7}$$

where N_1 is the initial number of particle windows. The empirical values of m and γ are 5 and 0.44, respectively. The exponential rule of (7) makes the number N_i decrease from stage to stage. Representative values of N_i are given in Table 1.

There are three problems with the MPW algorithm:

1) N_1 has to be large enough so that the N_1 particle windows to some extent overlap the objects in the image. Otherwise, $N_2, \ldots, N_m$ particle windows in later stages are hard to detect the objects. Extremely, if none of the number N_1 of particle windows are positives, then the subsequent N_2 new particle windows sampled from $q_1(\mathbf{w}) = \sum_{i=1}^{N_1} f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$ will not contain any clue of the location about the objects because $G(\mathbf{w}_i, \Sigma)$ is meaningless in this case.
2) MPW generates too many unnecessary particle windows around the object and object-like regions. Considering that classification of these regions is more time-consuming than obviously non-object regions in the algorithm of cascade AdaBoost, generating too many particle windows around the object and object-like regions greatly limits its efficiency.
3) The rule of parameter selection is not guaranteed to be optimal, because there is no reason to support that the values in Table 1 are the best.

IV. IMPROVED PARTICLE WINDOWS (IPW)

In this section, we propose to improve MPW in order to detect the objects in an image with a smaller number of particle windows (PWs).

Firstly, we define several concepts: rejection particle window, acceptance particle window, and ambiguity particle window. Secondly, the concepts of regions of rejection and acceptance are introduced. Finally, we will describe iPW algorithm based on these concepts and explain why iPW is superior to MPW.

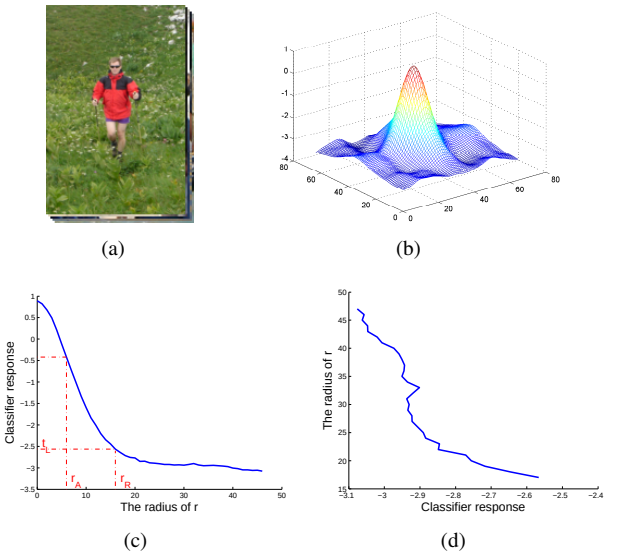
A. Rejection, Acceptance, and Ambiguity Windows

As stated in Section 3, each particle window represents a window $\mathbf{w} = (x, y, s)^T$ where x, y and s are the horizontal position, the vertical position and the size of the window, respectively. The particle window can also be expressed as

$\mathbf{w}(x, y, s)$. In the following, we use $\mathbf{w}(x, y)$ to represent $\mathbf{w}(x, y, s)$ when s is fixed.

We employ the classifier response $f(\mathbf{w})$ and its low and high thresholds (i.e., t_l and t_h) to define rejection, acceptance, and ambiguity particle windows, respectively:

1) A window $\mathbf{w}$ is called **Rejection PW** (**RPW**) if $f(w) < t_l$. Rejection PW is the particle window which can be definitely classified as negative class due to its low classifier response.
2) A window $\mathbf{w}$ is called **Acceptance PW (APW)** if $f(\mathbf{w}) \geq t_h$. Acceptance PW is the particle window which can be safely classified as positive class (object) because of its high value of classifier response.
3) A window $\mathbf{w}$ is called **Ambiguity PW** (**ABPW**) if $t_l \leq f(\mathbf{w}) < t_h$. One cannot classify ambiguity PW as positive class because its classifier response is not large enough, meanwhile cannot classify it as negative class because its classifier response is not low enough. And we call the set of the ambiguity particle windows $\mathbf{W}_{AB}$.



(a) (b) (c) (d)

Fig. 2. (a) The original images. (b) The average classifier response $f(\mathbf{w}_i)$ of the windows. (c) A profile of (b). (d) The rejection radius r_R varies with $f(\mathbf{w}_i)$.

B. Regions of Rejection and Acceptance

Region of Rejection (RoR) If a particle window $\mathbf{w}$ is considered as a rejection PW, it can be used to securely reject a set of nearby windows. The centers of the nearby windows $\mathbf{w}_i$ and $\mathbf{w}$ itself are called Region of Rejection (RoR) of $\mathbf{w}$. We denote the RoR by $\mathbf{R}_R$:

$$\mathbf{R}_R(\mathbf{w}) = \{x, y|\ ||\mathbf{w}_i - \mathbf{w}|| < r_R\}, \tag{8}$$

where the radius r_R is the maximum radius

$$r_R = \max_{\mathbf{w}_i} ||\mathbf{w}_i - \mathbf{w}||, \quad s.t.\ f(\mathbf{w}_i) < t_l. \tag{9}$$

r_R is a function of the classifier response $f(\mathbf{w})$. In Fig. 2(d), the smaller $f(\mathbf{w})$ is, the larger r_R is.

All the windows belonging to $\mathbf{R}_R(\mathbf{w})$ are represented as $\mathbf{W}_R(\mathbf{w})$.

Assumption 1 (Rejection Assumption). *If a window* $\mathbf{w}$ *is classified as a rejection window due to* $f(\mathbf{w}) < t_l$ *then all the windows* $\mathbf{W}_R(\mathbf{w})$ *can be directly rejected (i.e., labeled as negatives) without the necessity of computing the classifier response* $f(\mathbf{w}_i), \mathbf{w}_i \in \mathbf{W}_R(\mathbf{w})$.

This assumption is illustrated in Fig. 3(c), the red part in the image consists of regions of rejection.

Region of Acceptance (RoA) If a particle window $\mathbf{w}$ is considered as an acceptance PW, it can be used to securely accept a set of nearby windows. The centers of the nearby windows $\mathbf{w}_i$ and $\mathbf{w}$ itself form Region of Accept (RoA) of $\mathbf{w}$. Mathematically, RoA can be denoted by $\mathbf{R}_A$:

$$\mathbf{R}_A(\mathbf{w}) = \{x, y|\ ||\mathbf{w}_i - \mathbf{w}|| < r_A\}, \tag{10}$$

where the radius r_A is the maximum radius

$$r_A = \max_{\mathbf{w}_i} ||\mathbf{w}_i - \mathbf{w}||, \quad s.t.\ f(\mathbf{w}_i) \geq t_l. \tag{11}$$

Note that the constant is $f(\mathbf{w}_i) \geq t_l$ instead of $f(\mathbf{w}_i) \geq t_h$. Obviously, r_A with $f(\mathbf{w}_i) \geq t_l$ is larger than r_A with $f(\mathbf{w}_i) \geq t_h$.

All the windows belonging to $\mathbf{R}_A(\mathbf{w})$ are represented as $\mathbf{W}_A(\mathbf{w})$.

Assumption 2 (Acceptance Assumption). *If a window* $\mathbf{w}$ *is classified as acceptance window due to* $f(\mathbf{w}) \geq t_h$ *then all the windows* $\mathbf{W}_A(\mathbf{w})$ *should be directly accepted (i.e., labeled as positives) without the necessity of computing the classifier response* $f(\mathbf{w}_i), \mathbf{w}_i \in \mathbf{W}_A(\mathbf{w})$.

In Fig. 3(d), the blue part in the image consists of the regions of acceptance.

Let the center of a window $\mathbf{w}$ coincide with the center of an object and the size (scale) of the windows match that of the object very well (Fig. 2(a)). Then we compute the classifier responses of the neighboring windows $\mathbf{w}_i$. As shown in Fig. 2(b), it is usual that $f(\mathbf{w})$ is the largest and $f(\mathbf{w}_i)$ decreases with the distance $||\mathbf{w}_i - \mathbf{w}||$. In Fig. 2(c), the radius of r_A is chosen so that it satisfies $f(\mathbf{w}_i) \geq t_l$. Note that in Fig. 2(d), the rejection radius r_R is derived from Fig. 2(c).

C. iPW: Rejection-based Random Sampling

In this section, we describe iPW based on the concepts of RPW, APW, and ABPW, and their corresponding RoR and RoA.

1) Proposal Distribution of iPW: We introduce the motivation of iPW by firstly analyzing the properties and limitations of MPW.

Rejection Particle Windows and Dented Uniform Distribution $\tilde{u}_R(\mathbf{w})$ As discussed in Section 3.1, the particle windows of MPW are sampled from $\sum f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$. The contribution of a particle windows $\mathbf{w}_i$ is determined

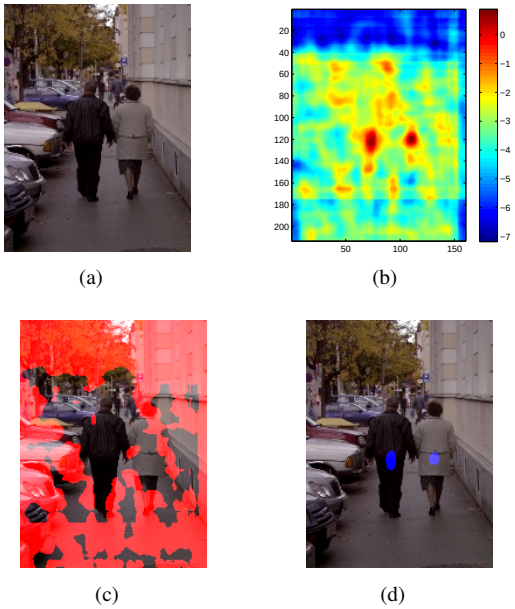


Fig. 3. Illustration of RoR and RoA. (a) The original image. (b) The classifier response. (c) Regions of rejection (RoR). (d) Regions of acceptance (RoA).

by its weight $f(\mathbf{w}_i)$. If $f(\mathbf{w}_i)$ is very small, the $\mathbf{w}_i$ contributes little to object detection because subsequent stage will not sample windows from the corresponding distribution $f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$. However, if only a small number of particle windows is generated, most of them will have small weights and the regions of objects will be not sampled . In this paper, we proposed how to make use of these particle windows with small weights (i.e., RPWs) for efficient object detection. One of the main contributions of the paper is to use rejection particle windows $\mathbf{w}_i$ and the corresponding $\mathbf{W}_R(\mathbf{w}_i)$ (i.e., the set of windows insides the region $\mathbf{R}_R(\mathbf{w}_i)$) to form a dented uniform distribution $\tilde{u}_R(\mathbf{w})$. Let the number of RPWs be N_R, then $\tilde{u}_R(\mathbf{w})$ is expressed as:

$$\tilde{u}_R(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \{\mathbf{W}_R(\mathbf{w}_1) \cup \ldots \cup \mathbf{W}_R(\mathbf{w}_{N_R})\}, \\ \frac{1}{a_R}, & \text{otherwise}, \end{cases} \tag{12}$$

where a_R satisfies $\iiint \frac{1}{a_R} dxdyds = 1$. Fig. 4(b) illustrates an one-dimensional dented uniform distribution. Obviously, sampling from this dented uniform distribution avoids drawing windows from the existing regions of rejection.

Acceptance Particle Window and Dented Uniform Distribution $\tilde{u}_A(\mathbf{w})$ In MPW, when a particle window $\mathbf{w}_i$ (i.e., acceptance particle window) has large weight $f(\mathbf{w}_i)$, it has large contribution to the distribution $\sum f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma)$. Sampling from this updated distribution will generate particle windows overlapping or even coinciding with $\mathbf{w}_i$. We think that it is redundant to resample the acceptance window $\mathbf{w}_i$ and its close neighbors. To avoid the redundancy, we propose to employ the acceptance particle windows to maintain and update a dented uniform

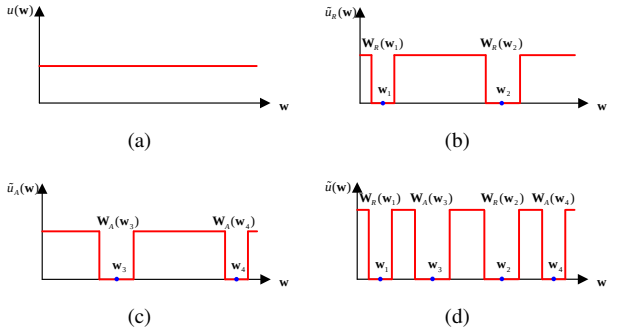


Fig. 4. Uniform distribution and dented uniform distribution. (a) Uniform distribution. (b) Dented uniform distribution $\tilde{u}_R(\mathbf{w})$ formed by two rejection particle windows $\mathbf{w}_1$ and $\mathbf{w}_2$. (c) Dented uniform distribution $\tilde{u}_A(\mathbf{w})$ formed by two acceptance particle windows $\mathbf{w}_3$ and $\mathbf{w}_4$. (d) Mixture dented unform distribution by combing the rejection and acceptance particle windows.

distribution $\tilde{u}_A(\mathbf{w})$:

$$\tilde{u}_A(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \{\mathbf{W}_A(\mathbf{w}_1) \cup \ldots \cup \mathbf{W}_A(\mathbf{w}_{N_A})\}, \\ \frac{1}{a_A}, & \text{otherwise}, \end{cases} \tag{13}$$

where a_A satisfies $\iiint \frac{1}{a_A} dxdyds = 1$, and N_A is the number of acceptance particle windows. Fig. 4(c) illustrates a $\tilde{u}_A(\mathbf{w})$.

Obviously, both the rejection and acceptance particle windows play role in excluding regions in uniform distribution. Therefore, as illustrated in Fig. 4(d), we propose to combine $\tilde{u}_R(\mathbf{w})$ and $\tilde{u}_A(\mathbf{w})$ into a unified dented uniform distribution $\tilde{u}(\mathbf{w}) = \tilde{u}_R(\mathbf{w}) \times \tilde{u}_A(\mathbf{w})$:

$$\tilde{u}(\mathbf{w}) = \begin{cases} 0, & \mathbf{w} \in \{\mathbf{W}_R \cup \mathbf{W}_A\}, \\ \frac{1}{a}, & \text{otherwise}, \end{cases} \tag{14}$$

where a satisfies $\iiint \frac{1}{a} dxdyds = 1$.

Ambiguity Particle Windows and Dented Gaussian Distribution Suppose that there are N_{AB} ambiguity particle windows in $\mathbf{W}_{AB}$. With the class label being either positive or negative, the region nearby the ambiguity particle window exhibits the largest uncertainty relative to the rejection particle windows and the acceptance particle windows, so it contains potential clue for objects. One way to exploit the N_{AB} ambiguity particle windows is directly using them for modeling a mixture of Gaussian distribution $g(\mathbf{w})$:

$$g(\mathbf{w}) = \sum_{i=1}^{N_{AB}} f(\mathbf{w}_i)G(\mathbf{w}_i, \Sigma). \tag{15}$$

However, our experimental results show that sampling from $g(\mathbf{w})$ may result in particle windows overlapping with the existing rejection or acceptance particle windows. Clearly, it is useless to sample such particle windows. Therefore, to remove the redundancy, we propose to model a mixture of dented Gaussian distribution $\tilde{g}(\mathbf{w})$ using the N_{AB} ambiguity particle windows with the help of the existing rejection and acceptance particle windows or equivalently the current dented uniform distribution $\tilde{u}(\mathbf{w})$:

$$\tilde{g}(\mathbf{w}) = \sum_{i=1}^{N_{AB}} f(\mathbf{w}_i) \left[G(\mathbf{w}_i, \Sigma) \times (a \times \tilde{u}(\mathbf{w}))\right]. \tag{16}$$

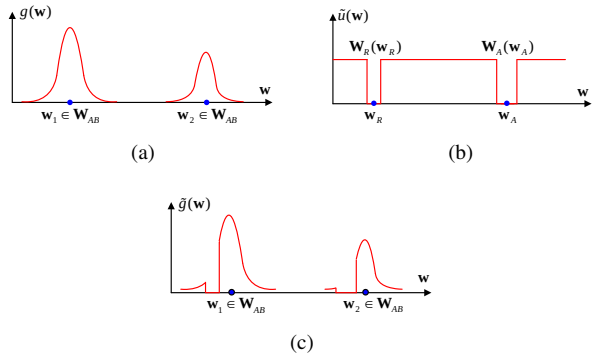


Fig. 5. Dented Gaussian distribution. (a) Gaussian distribution. (b) Dented uniform distribution. (c) Mixture of dented Gaussian distribution.

Because $a \times \tilde{u}(\mathbf{w}) = 0$ in the regions of rejection and acceptance and $a \times \tilde{u}(\mathbf{w}) = 1$ elsewhere, so multiplying $G(\mathbf{w}_i, \Sigma)$ with $a \times \tilde{u}(\mathbf{w})$ results in dented Gaussian distribution as is illustrated in Fig. 5. In Fig. 5(a) two ambiguity particle windows $\mathbf{w}_1$ and $\mathbf{w}_2$ in $\mathbf{W}_{AB}$ form a mixture of Gaussian distribution $g(\mathbf{w})$. In Fig. 5(b) a rejection particle window $\mathbf{w}_R$ and an acceptance particle window $\mathbf{w}_A$ form a dented uniform distribution $\tilde{u}(\mathbf{w})$. Fig. 5(c) shows the final dented Gaussian ditribution $\tilde{g}(\mathbf{w})$.

Proposal Distribution. The proposal distribution $\tilde{q}(\mathbf{w})$ of iPW is a weighted average of the dented uniform distribution $\tilde{u}(\mathbf{w})$ and mixture of dented Gaussian distribution $\tilde{g}(\mathbf{w})$:

$$\begin{aligned}\tilde{q}(\mathbf{w}) &= P_u \times \tilde{u}(\mathbf{w}) + P_g \times \tilde{g}(\mathbf{w}) \\ &= P_u \times \tilde{u}(\mathbf{w}) \\ &+ P_g \sum_{i=1}^{N_{AB}} f(\mathbf{w}_i) \left[G(\mathbf{w}_i, \Sigma) \times (a \times \tilde{u}(\mathbf{w}))\right].\end{aligned} \tag{17}$$

More generally, the proposal distribution $\tilde{q}_i(\mathbf{w})$ in stage i is expressed as

$$\begin{aligned}\tilde{q}_i(\mathbf{w}) &= P_u(i) \times \tilde{u}_i(\mathbf{w}) \\ &+ P_g(i) \sum_{j=1}^{N_{AB}} f(\mathbf{w}_j) \left[G(\mathbf{w}_j, \Sigma) \times (a \times \tilde{u}_i(\mathbf{w}))\right].\end{aligned} \tag{18}$$

The weights P_u and P_g can be regarded as the posterior probabilities for $\mathbf{w}_i$ to be generated from $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$, respectively. That is $p(\tilde{u}(\mathbf{w})|\mathbf{w}_i) = P_u$ and $p(\tilde{g}(\mathbf{w})|\mathbf{w}_i) = P_g$ which can be respectively defined by

$$P_u = \alpha \times (1 - \frac{N_R + N_A}{N}), \text{ and } P_g = 1 - P_u. \tag{19}$$

In (19), $\alpha \in [0, 1]$ is used for performance adjusting, $N_R = |\mathbf{W}_R|$ and $N_A = |\mathbf{W}_A|$ are the numbers of rejection and acceptance particle windows in $\mathbf{W}_R$ and $\mathbf{W}_A$, respectively.

Hypothesis 1. Suppose there are two stages and the number of particle windows in stage 1 and stage 2 are N_1 and N_2, respectively. In both MPW and iPW, the N_1 particle windows are generated from the same uniform distribution. But the N_2 particle windows in stage 2 of MPW are sampled from $q(\mathbf{w})$ whereas they are sampled from $\tilde{q}(\mathbf{w})$ in iPW. Then the probability for N_2 particle windows in iPW to contain the object is larger than that in MPW.

It is trivial to prove Hypothesis 1. If there are non-zero number of rejection and/or acceptance particle windows, then the search region is reduced by these particle windows (equivalently, the dented uniform distribution $\tilde{u}_1(\mathbf{w})$). Consequently, sampling the same number of particle windows from reduced search domain is better than from the original large domain in the sense of detecting the objects. Generally, if both iPW and MPW use the same number of particle windows in each stage, then the probability for iPW to detect the objects is larger than that of MPW.

Each Stage Consists of One Particle Window So far, we have designed the new proposal distribution of iPW. The question is that how many particle windows are to be generated in each stage. In MPW, the exponential rule of (7) is used for setting the window number. But this is far from optimal. Intuitively, we think that it is optimal if each stage contains one particle window. However, it fails completely for MPW. Throughout the paper, iPW means the one where each stage has a single new particle window. The number of generated particle windows incrementally increases one by one. The first letter 'i' of "iPW" is named after "incremental".

2) Basic Algorithm of iPW: The core of iPW is iteratively sampling particle windows from the proposal distribution $\tilde{q}(\mathbf{w})$ and updating the sets of rejection, acceptance, and ambiguity particle windows. Because the proposal distribution $\tilde{q}(\mathbf{w})$ is a weighted average of the dented distributions $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$, drawing a particle window from $\tilde{q}(\mathbf{w})$ is equivalent to drawing from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$ with the probabilities P_u and P_g, respectively.

The basic algorithm of iPW is given in Algorithm 2. The output is $\mathbf{W}_P$ (the final set of positive particle windows) on which non-maximum-suppression is applied for final object detection.

In the initialization step, the sets of rejection, acceptance, and ambiguity particle windows are emptied (line 2). The dented uniform distribution $\tilde{u}(\mathbf{w})$ is initialized by the uniform distribution $u(\mathbf{w}) = 1/N$ because currently there are no rejection and acceptance particle windows (line 3). The mixture of dented Gaussian distribution $\tilde{g}(\mathbf{w})$ is initialized to be 0 because so far there are no ambiguity particle windows to really construct it (line 3).

In each iteration, a particle window is generated from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$ until the predefined number N_{iPW} of particle windows is obtained. According to the value $f(\mathbf{w})$ of classifier response, the generated particle window $\mathbf{w}$ is classified as rejection, acceptance, or ambiguity particle window. If $\mathbf{w}$ is classified as rejection particle window, then all the windows $\mathbf{W}_R(\mathbf{w})$ in the $\mathbf{R}_R(\mathbf{w})$ (region of rejection particle window of $\mathbf{w}$) will be remerged into $\mathbf{W}_R$. If $\mathbf{w}$ is classified as acceptance particle window, then all the windows $\mathbf{W}_A(\mathbf{w})$ in the $\mathbf{R}_A(\mathbf{w})$ (region of acceptance particle window of $\mathbf{w}$) will be remerged into $\mathbf{W}_A$. Otherwise it will be put into the window set $\mathbf{W}_{AB}$ (see line 9, 10 and 11).

Algorithm 2 The basic algorithm of iPW.

Input:

The number N of all candidate windows;
The number N_{iPW} of total particle windows;
High and low classifier thresholds t_h and t_l, respectively;

Output:

The set $\mathbf{W}_P$ of positive particle windows.

1: **Initialization**
2: Empty the sets of rejection, acceptance, and ambiguity particle windows: $\mathbf{W}_R \leftarrow \Phi$, $\mathbf{W}_A \leftarrow \Phi$, and $\mathbf{W}_{AB} \leftarrow \Phi$, respectively. Let the numbers of rejection, acceptance, and ambiguity particle windows be $N_R = 0$, $N_A = 0$, and $N_{AB} = 0$.
3: Initialize the dented uniform and dented Gaussian distributions by $\tilde{u}(\mathbf{w}) \leftarrow 1/N$ and $\tilde{g}(\mathbf{w}) \leftarrow 0$.
4: **Iteration:**
5: **for** $i = 1$ **to** N_{iPW} **do**
6: $P_u = \alpha \times (1 - \frac{N_A + N_R}{N})$, $P_g = 1 - P_u$.
7: Sample a particle window $\mathbf{w}$ from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$. The probabilities for $\mathbf{w}$ to be generated from $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$ are P_u and P_g, respectively.
8: Put $\mathbf{w}$ into $\mathbf{W}_R$, $\mathbf{W}_A$, or $\mathbf{W}_{AB}$ according to the classifier response:
9: If $f(\mathbf{w}) < t_l$, then $\mathbf{W}_R = \mathbf{W}_R \cup \mathbf{W}_R(\mathbf{w})$, $N_R \leftarrow |\mathbf{W}_R|$;
10: If $f(\mathbf{w}) \geq t_h$, then $\mathbf{W}_A = \mathbf{W}_A \cup \mathbf{W}_A(\mathbf{w})$, $N_A \leftarrow |\mathbf{W}_A|$, and $\mathbf{W}_P = \mathbf{W}_P \cup \mathbf{w}$;
11: If $t_l \leq f(\mathbf{w}) < t_h$, then $\mathbf{W}_{AB} = \mathbf{W}_{AB} \cup \mathbf{w}$, $N_{AB} \leftarrow |\mathbf{W}_{AB}|$.
12: Update $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$ using the updated $\mathbf{W}_R$, $\mathbf{W}_A$, and $\mathbf{W}_{AB}$.
13: **end for**
14: **return** $\mathbf{W}_P$.

Finally, based on the updated $\mathbf{W}_R$, $\mathbf{W}_A$, and $\mathbf{W}_{AB}$, the dented distributions $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$ are updated according to (14) and (16), respectively.

Characteristics of the iPW algorithm Fig. 6 demonstrates the characteristics of the iPW algorithm. Because each stage (iteration) generates a single particle window, the cumulative number $N_{iPW}(i)$ of generated particle windows at stage i is $N_{iPW}(i) = i$. Among the $N_{iPW}(i)$ particle windows, $P_u(i)$ fraction is sampled from $\tilde{u}(\mathbf{w})$ whereas $P_g(i)$ fraction is sampled from $\tilde{g}(\mathbf{w})$. That is, the number of windows coming from $\tilde{u}(\mathbf{w})$ is $N^u_{iPW}(i) = P_u(i) \times N_{iPW}(i)$, and the number of windows coming from $\tilde{g}(\mathbf{w})$ is $N^g_{iPW}(i) \approx P_g(i) \times N_{iPW}(i)$. Fig. 6(a) shows that most of the generated particle windows are from $\tilde{u}(\mathbf{w})$ in the first several stages. But its fraction (i.e., $P_u(i)$) decreases as iteration proceeds meanwhile the fraction $P_g(i)$ increases (see Fig. 6(b)).

The above phenomenon is explained as follows. Because the number of object windows is very small relative to the total number of windows in the image, drawing a particle window from the initial distribution $u(\mathbf{w})$ and

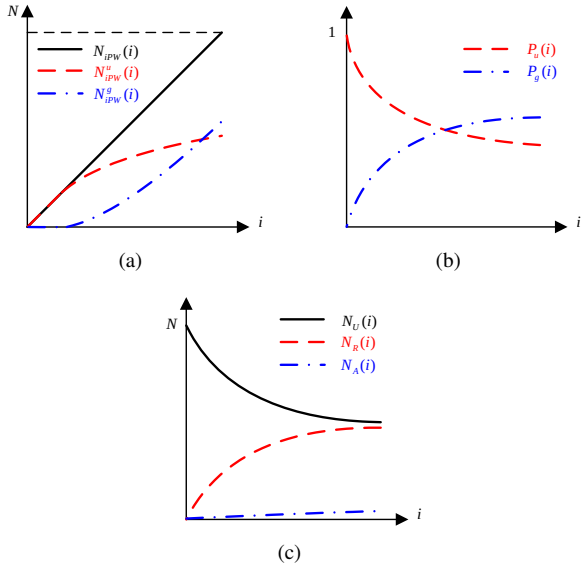


Fig. 6. Characteristics of iPW. (a) The curves of $N_{iPW}(i)$, $N^g_{iPW}(i)$, and $N^u_{iPW}(i)$. (b) The curves of $P_u(i)$ and $P_g(i)$. (c) The curves of $N_U(i)$, $N_R(i)$, and $N_A(i)$.

the distribution $\tilde{u}_i(\mathbf{w})$ in first few stages (e.g., $i = 10$) will result in rejection particle windows in a very large probability. Consequently, the number $N_R(i)$ of rejection particle windows increases very fast with i, but the number $N_A(i)$ of acceptance particle windows increases very slowly (see Fig. 6(c)), which makes the number $N_u(i) = N - N_R(i) - N_A(i)$ of unvisited windows be very large for small i. According to (19), $P_u(i) \gg P_g(i)$. But as iteration proceeds, $N_R(i)$ and $N_A(i)$ increase monotonically making P_u decrease.

As $N_R(i)$ and $N_A(i)$ increases monotonically with i, so it can pay more attention to the other unvisited potential areas. This characteristic makes iPW not to generate unnecessary too many particle windows around the object and object-like regions. It is known that classification of these regions is very time-consuming if cascade AdaBoost is adopted. This is one of the advantages of iPW over MPW.

3) Semi-incremental Version of iPW (siPW): Algorithm 2 is a purely incremental algorithm which has some problems. First, in the first few iterations the number of particle windows is very small, so $N_{AB}(i)$ in it is much smaller. In this case, $\tilde{g}(\mathbf{w})$ can't reflect the probability distribution of whole image. In order to have enough ambiguity particles to represent the probability distribution, the variables N^*_C and b are introduced in Algorithm 3. Through them, the particle windows are forcibly sampled from $\tilde{u}(\mathbf{w})$ in first several stages until that a certain number N^*_C of particle windows, especially the ambiguity particle windows, is available. By this way, it can better reflect the probability of whole image. Second, if a particle window is sampled from $\tilde{g}(\mathbf{w})$ and then $\tilde{g}(\mathbf{w})$ is immediately updated, the latter sampled particle windows will be heavily centered on the strongest classifier response regions. Namely, the regions with strongest responses will be enhanced more and more, while the regions with the relative lower responses

Algorithm 3 Semi-incremental version of iPW.

Input:

The number N of all candidate windows;
The number N_{iPW} of total particle windows;
High and low classifier thresholds t_h and t_l, respectively;
The threshold number N_C^* for the number of ambiguity particle windows

Output:

The set $\mathbf{W}_P$ of positive particle windows.

1: **Initialization**
2: Empty the sets of rejection, acceptance, and ambiguity particle windows: $\mathbf{W}_R \leftarrow \Phi$, $\mathbf{W}_A \leftarrow \Phi$, and $\mathbf{W}_{AB} \leftarrow \Phi$, respectively. Let the numbers of rejection, acceptance, and ambiguity particle windows be $N_R = 0$, $N_A = 0$, and $N_{AB} = 0$.
3: Initialize the dented uniform and dented Gaussian distributions by $\tilde{u}(\mathbf{w}) \leftarrow 1/N$ and $\tilde{g}(\mathbf{w}) \leftarrow 0$.
4: *Initialize the binary indicator $b = 0$ and the number of cumulative particle windows $N_C = 0$.*
5: **Iteration:**
6: **for** $i = 1$ **to** N_{iPW} **do**
7: If $b = 0$, then $P_u \leftarrow 1$ and $P_g \leftarrow 0$, else $P_u = a \times (1 - \frac{N_A + N_R}{N})$ and $P_g = 1 - P_u$.
8: Sample a particle window $\mathbf{w}$ from either $\tilde{u}(\mathbf{w})$ or $\tilde{g}(\mathbf{w})$ with P_u and P_g, respectively. $N_C = N_C + 1$.
9: Put $\mathbf{w}$ into $\mathbf{W}_R$, $\mathbf{W}_A$, or $\mathbf{W}_{AB}$ according to the classifier response:
10: If $f(\mathbf{w}) < t_l$, then $\mathbf{W}_R = \mathbf{W}_R \cup \mathbf{W}_R(\mathbf{w})$, $N_R \leftarrow |\mathbf{W}_R|$;
11: If $f(\mathbf{w}) \geq t_h$, then $\mathbf{W}_A = \mathbf{W}_A \cup \mathbf{W}_A(\mathbf{w})$, $N_A \leftarrow |\mathbf{W}_A|$, and $\mathbf{W}_P = \mathbf{W}_P \cup \mathbf{w}$;
12: If $t_l \leq f(\mathbf{w}) < t_h$, then $\mathbf{W}_{AB} = \mathbf{W}_{AB} \cup \mathbf{w}$, $N_{AB} \leftarrow |\mathbf{W}_{AB}|$.
13: Update $\tilde{u}(\mathbf{w})$ using the updated $\mathbf{W}_R$ and $\mathbf{W}_A$.
14: *If $N_C = N_C^*$, then $b = 1$, update $\tilde{g}(\mathbf{w})$, $\mathbf{W}_{AB} \leftarrow \Phi$, $N_C^* = N_C^* \times e^{-\gamma}$, and $N_C = 0$.*
15: **end for**
16: **return** $\mathbf{W}_P$.

where object exists will be ignored. So instead of updating $\tilde{g}(\mathbf{w})$ per particle window, it is wise to update $\tilde{g}(\mathbf{w})$ until there is a certain number of particle windows (line 14). To overcome the above problems, a semi-incremental version of iPW (i.e., Algorithm 3) is proposed. The main differences from Algorithm 2 are written in italic. We call it semi-incremental algorithm because $\tilde{g}(\mathbf{w})$ is updated once there is a certain number N_C^* of pariticle windows, though each stage has one particle window and the rejection regions are updated in purely incremental manner.

4) Efficiently Sampling From Dented Uniform and Gaussian Distributions: As can be seen from (1), the computation time of an object detection algorithm is composed of the time of window generation, feature extraction, and classification. So it is important for iPW to efficiently generating particle windows from $\tilde{u}(\mathbf{w})$ and $\tilde{g}(\mathbf{w})$.

To efficiently draw a particle window from $\tilde{u}(\mathbf{w})$, we, in Algorithm 4, propose to iteratively draw a window from standard uniform distribution $u(\mathbf{w})$ until it does not belong to $\mathbf{W}_R$ or $\mathbf{W}_A$. Similarly, to efficiently draw a particle window from $\tilde{g}(\mathbf{w})$, in Algorithm 4 we propose to iteratively draw a window from standard mixture of Gaussian distribution $g(\mathbf{w})$ until it does not coincide with the elements of $\mathbf{W}_R$ and $\mathbf{W}_A$. As one can design algorithm for checking $\mathbf{w} \in \mathbf{W}_R$ and $\mathbf{w} \in \mathbf{W}_A$ in an extremely efficient manner, the computation time of window generation in iPW is negligible. The maximum iterations number $N_{\max}$ is used for avoiding infinite loops.

Algorithm 4 Draw a particle window $\mathbf{w}$ from $\tilde{u}(\mathbf{w})$ (or $\tilde{g}(\mathbf{w})$).

Input:

The sets of rejection and acceptance particle windows: $\mathbf{W}_R$ and $\mathbf{W}_A$, respectively;
The maximum iteration number $N_{\max}$;

Output:

a particle window $\mathbf{w}$.

1: **for** $n = 1$ **to** $N_{\max}$ **do**
2: Draw a window $\mathbf{w}_n$ from the uniform distribution $u(\mathbf{w})$ (or $g(\mathbf{w})$).
3: If $\mathbf{w} \notin \mathbf{W}_R$ and $\mathbf{w} \notin \mathbf{W}_A$, then $\mathbf{w} = \mathbf{w}_n$, and break.
4: **end for**
5: **return** $\mathbf{w}$.

V. Experimental Results

A. Experimental Setup

Experiments are carried out on the INRIA pedestrian dataset and the MIT-CMU face dataset to compare the proposed iPW with MPW and SW. To detect pedestrians in INRIA dataset, HOG and SVM [12] are used for features and classifier, respectively. Haar-like features and cascade AdaBoost classifier are employed for detecting faces in the MIT-CMU dataset [41]. The source code is publicly accessible at http://yanweipang.com/papers.

Intermediate results are also given to show the rationality of the assumptions mentioned before.

B. Results on the INRIA Pedestrian Database.

In the INRIA dataset, the positive training set consists of 1208 normalized pedestrian windows, and the negative training set contains a mass of windows sampled from 1218 big and non-pedestrian images. The image size of the training window is 128×64 pixels, from which a 3780-dimensioned HOG feature vector is extracted. A linear SVM classifier $f(\mathbf{w})$ is obtained from the training sets.

As can be seen from Algorithms 2 and 3, the explicit parameters of iPW are t_l, t_h, N_C^*, α and γ. In our experiments, $t_l = -2.0$ and $t_h = 0$ are used. Because rejection and acceptance particle windows are defined by not only t_l and t_h, but also r_R and r_A. So the parameters also include r_R and r_A. In (8) and (10), the regions of rejection and acceptance are circular and isotropic whose

TABLE III
DETECTION RATES VARY WITH THE NUMBER N OF PARTICLE WINDOWS WHEN FPPI = 0.1.

N	2367	7100	11833	16567	21300	26033
MPW	0.469	0.594	0.614	0.623	0.625	0.627
iPW	0.561	0.620	0.625	0.627	0.628	0.630

size are determined by r_R and r_A, respectively. However, because the height h of the pedestrian is larger than its width w, it is more reasonable that the region of rejection and acceptance is rectangle. The size of the rectangle is represented as $r_R^x \times r_R^y$. Likewise, the size of region of acceptance is represented as $r_A^x \times r_A^y$. As stated in Section 4.2, r_R^x and r_R^y depend on the classifier response $f(\mathbf{w})$. In our experiments, r_R^x and r_R^y are quantized to 9 intervals according to the value of $f(\mathbf{w})$. r_R^x and r_R^y also depend on the object width w and height h in question. Solid experiments are conducted to find rule for setting r_R^x and r_R^y according to $f(\mathbf{w})$, h and w. Table 2 shows how to choose r_R^x and r_R^y. Note that $h = 128$ and $w = 64$. In Algorithm 3, we only use r_R^x and r_R^y when $f(\mathbf{w})$ belongs to the first four intervals. r_A^x/w and r_A^y/h are set to 0.16 and 0.16, respectively. $N_C^* = 0.5N_{iPW}$ is employed in the initialization step of Algorithm 3. The parameters α and γ are set 0.2 and 0.7, respectively.

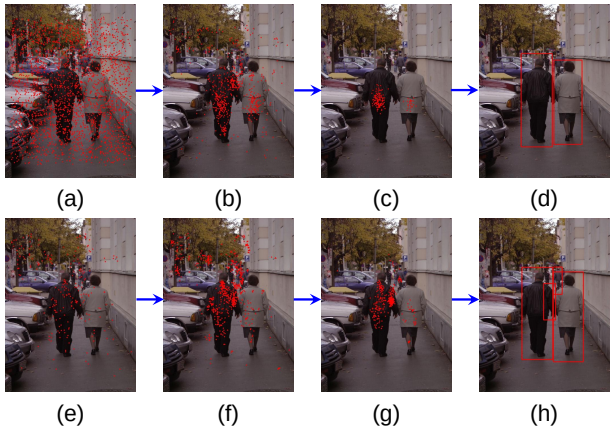
It is noted that regions of rejection and acceptance are cubic when scale factor is considered. The testing image is zoomed out by a factor $1/1.05$. If a window is rejected at current scale s, which belongs to the interval $N_{Interval}$, then the windows in adjacent scales s' from $s \times 1.05^{3-N_{Interval}}$ to $s/1.05^{3-N_{Interval}}$ with the size $0.8^{\Delta} r_R^x \times 0.8^{\Delta} r_R^y$ ($\Delta = |\log_{1.05}^{s/s'}|$) are also rejected. If a window is accepted at current scale s, then the windows in adjacent scales s' from $s \times 1.05^3$ to $s/1.05^3$ with the size $0.8^{\Delta} r_A^x \times 0.8^{\Delta} r_A^y$ ($\Delta = |\log_{1.05}^{s/s'}|$) are also accepted.

Table 3 compares iPW with MPW in terms of detection rate when they generate and examine the same number N of particle windows. When $N = 2367$, the detection rate of iPW is 0.561 whereas the detection rate of MPW is 0.469, meaning that the detection rate of iPW is 9.2% higher than that of MPW. As N decreases, the advantage of iPW becomes more remarkable.

To further see the advantage of iPW, we show in Fig. 7 a specific detection result of iPW and MPW when N is as small as 500. The red dots in Fig. 7(a) indicate the centers of particle windows in the last stage of MPW. Fig. 7(b) gives the final detection result by non-maximum suppression, where the man is detected whereas the woman is missed. The ambiguity particle windows of the last stage of iPW are shown in Fig. 7(c) and the final detection result is shown in Fig. 7(d). On the one hand, Fig. 7 demonstrates that MPW fails to detect the woman when a small number of particle windows is sampled whereas iPW is capable to localize both the woman and man. iPW generates the particle windows one by one. If the generated particle window has a lower classifier response, then the particle window (i.e., rejection particle window) will tell

(a) (b) (c) (d)

Fig. 7. The detection results of iPW and MPW under 500 particle windows. (a) The particle windows of the last stage in MPW. (b) Final detection result of MPW. (c) The particle windows of the last stage in iPW. (d) Final detection result of iPW.



(a) (b) (c) (d)
(e) (f) (g) (h)

Fig. 8. MPW generates unnecessary too many particle windows around the pedestrian regions. (a), (b) and (c) are the updating process of particle windows in MPW. (d) is the final detection result of MPW. (e), (f) and (g) are the updating process of particle windows in iPW. (h) is the final detection result of iPW.

the next particle window not to sample from it and its neighboring region (i.e., region of rejection). As a result, other regions including the object regions will have larger possibility to be investigated. By contrast, MPW does not have the rejection mechanism. When the current particle windows do not contain clues of the objects, it is almost impossible for the latter particle windows to capture the location information of the objects.

On the other hand, comparing Fig. 7(a) and (c), one can observe that MPW generates unnecessary too many particle windows around the man, whereas iPW can properly assign the limited number of particle windows to both the man region and the woman region. This phenomenon can be more clearly seen form Fig. 8. Fig. 8(a)-(c) give the updating process of particle windows in MPW, Fig. 8(d) shows that MPW is able to detect two persons if there is enough number of particle windows in the initialization. Fig. 8(e)-(g) give the updating process of particle windows in iPW, Fig. 8(h) shows that iPW detect even four pedestrians, including a false positive.

Fig. 9 shows how $N_v^{iPW}(i) = |\mathbf{W}_R| + |\mathbf{W}_A|$ and $N_u^{iPW}(i) = N - |\mathbf{W}_R| - |\mathbf{W}_A|$ vary with the number i of generated particle windows of iPW algorithm and how $N_v^{MPW}(i) = i$ and $N_u^{MPW}(i) = N - i$ vary with i of MPW algorithm. One can see that the number $N_v^{iPW}(i)$ of visited windows of iPW grows much faster than that of MPW.

TABLE II
SET r_R^x AND r_R^y ACCORDING TO $f(\mathbf{w})$, h AND w.

$N_{Interval}$	0	1	2	3	4	5	6	7	8
$f(\mathbf{w})$	[-inf, -4.0]	[-4.0, -3.5]	[-3.5, -3.0]	[-3.0, -2.5]	[-2.5, -2.0]	[-2.0, -1.5]	[-1.5, -1.0]	[-1.0, -0.5]	[-0.5, 0.0]
r_R^x/w	0.22	0.18	0.16	0.12	0.10	0.06	0.06	0.02	0.02
r_R^y/h	0.22	0.18	0.16	0.12	0.10	0.06	0.06	0.02	0.02

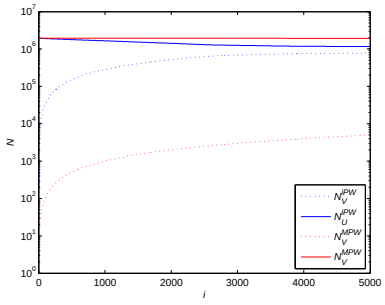


Fig. 9. The variations of N_v^{MPW}, N_u^{MPW}, N_v^{iPW} and N_u^{iPW}.

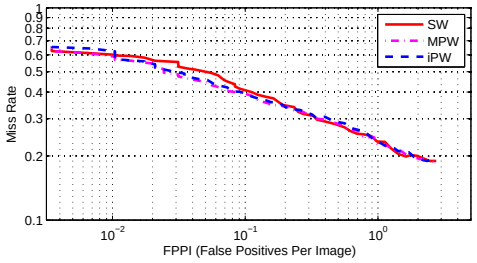


Fig. 10. DET curves comparing iPW with MPW and SW on INRIA.

Equivalently, the number $N_u^{iPW}(i)$ of unvisited windows of iPW drops much faster than that of MPW. So generating the same number of particle windows, iPW can classify (reject or accept) more windows (regions) than MPW. This explains why iPW obtains better detection accuracy than MPW when they use the same number of particle windows.

If a smaller number of particle windows is generated, can iPW achieve the same detection accuracy as MPW? If it is true, then one can conclude that iPW is more efficient than MPW. To answer this question, SW is used as a baseline. It scans the image with the pixel stride 8 and scaling factor 1.05, and the number of scanned windows is denoted by N_{SW}. N_{SW} varies with the size of testing image. For a 480×640 image, N_{SW} is 47335. Let MPW generate $N_{MPW} = 0.3 \times N_{SW}$ particle windows and iPW generate $N_{iPW} = 0.15 \times N_{SW}$ particle windows. The resulting curves of miss rate vs. FFPI are plotted in Fig. 10. It is seen that these different window generation algorithms have very close operating points. For example, the miss rates of SW, MPW and iPW are 23.4%, 22.8%, and 23.4%, respectively when FFPI=1. At these operating points, the average values of N_{SW}, N_{MPW} and N_{iPW} in INRIA are shown in Table 4. Table 4 shows that to achieve the same operating point SW has to investigate 47335 windows whereas it is enough for iPW to generate and check 7099 windows. The detection time T_{SW} of SW is 3.94 times of that (i.e., T_{iPW}) of iPW. Moreover, $N_{iPW}/N_{MPW} = 0.499$ means that using half of particle windows iPW can obtain the same detection rate as MPW. The ratio of detection time T_{MPW} of MPW and detection time T_{iPW} of iPW is 1.8, implying much higher efficiency of iPW than MPW.

C. Results on the MIT-CMU Face Database.

In Section 5.2, the feature and classifier are HOG and SVM, respectively. In this section, we evaluate iPW by using Haar-like features and cascade AdaBoost classifier for detecting faces in the standard MIT-CMU face database. The testing set consists of 125 images with 483 frontal faces. A 10 layers cascade model is learnt from 20000 normalized 20×20 small face images and 5000 non-face large negative images.

Because the range of response of cascade AdaBoost is quite different from SVM, the low and high classifier thresholds t_l and t_h are also different from those in Section 5.2. Specifically, $t_l = 0.2$ and $t_h = 1.0$ are adopted. Consequently, the length r_R and r_A of regions of rejection and acceptance should be tuned. r_R and r_A are related to $f(\mathbf{w})$, h and w. But the detection window is square, so $h = w$. The classifier response $f(\mathbf{w})$ of a cascade AdaBoost is defined by $f(\mathbf{w}) = j_{\mathbf{w}}/L$, where $j_{\mathbf{w}}$ is the index j of the last stage which provides a positive classification for $\mathbf{w}$, and $L = 10$ is total number of the stages of the cascade structure. The relationship between r_R, $f(\mathbf{w})$ and h is given in Table 5 where r_R/h monotonously decreases with $f(\mathbf{w})$. In Alogorithm 3, we only use the r_R when $f(\mathbf{w})$ belongs to the first two values. r_A^x/w and r_A^y/h are set to 0.1 and 0.1, respectively. $N_C^* = 0.5N_{iPW}$ is employed in the initialization step of Algorithm 3. The parameters α and γ are set 0.2 and 0.7, respectively.

Similar to Section 5.2, regions of rejection and acceptance are cubic. The testing image is zoomed out by a factor $1/1.15$. If a window is rejected at current scale s, the windows in neighboring scales s' from $s \times 1.15$ to $s/1.15$ with the size $0.5^{\Delta} r_R^x \times 0.5^{\Delta} r_R^y$ ($\Delta = |\log_{1.15}^{s/s'}|$) form the $\mathbf{W}_R$ of rejection particle windows. If a window is accepted at current scale s, then the windows in adjacent scales s' from $s \times 1.15$ to $s/1.15$ with the size $0.5^{\Delta} r_A^x \times 0.5^{\Delta} r_A^y$ ($\Delta = |\log_{1.15}^{s/s'}|$) are also accepted.

With the above parameters, iPW is applied to 125 testing images and detection rates corresponding to different number N of particle windows are shown in Table 6. Table 6 also gives the detection rates of MPW. It is observed that iPW has almost higher detection rate in each case. When

TABLE IV
EFFICIENCY OF SW, MPW AND IPW ON INRIA.

N_{SW}	N_{MPW}	N_{iPW}	N_{iPW}/N_{MPW}	T_{MPW}/T_{iPW}	T_{SW}/T_{MPW}	T_{SW}/T_{iPW}
47335	14200	7099	0.49	1.80	2.19	3.94

TABLE V
SET r_R ACCORDING TO $f(\mathbf{w})$ AND h.

$f(\mathbf{w})$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
r_R/h	0.100	0.090	0.060	0.050	0.050	0.040	0.040	0.030	0.040	0.030

TABLE VI
DETECTION RATES VARY WITH THE NUMBER N OF PARTICLE WINDOWS WHEN FPPI = 0.1.

N	25066	75200	125333	175466	225600	275733
MPW	0.583	0.758	0.788	0.806	0.811	0.812
iPW	0.676	0.792	0.806	0.813	0.816	0.819

(a) (b) (c) (d)

Fig. 11. The detection results of iPW and MPW under 5000 particle windows. (a) The particle windows of the last stage in MPW. (b) Final detection result of MPW. (c) The particle windows of the last stage in iPW. (d) Final detection result of iPW.

the number of particle windows is small, the advantage of iPW is more remarkable. For example, when $N = 25066$, the detection rate of iPW is 9.3% higher than that of MPW.

Fig. 11(b) and (d) respectively show the detection results of MPW and iPW when the number of particle windows is limited to 5000. Clearly, iPW is capable of detecting the two faces in the testing image whereas MPW does not detect any face at all. Fig. 11(a) and (c) show the centers of the particle windows generated in the last stage of MPW and iPW, respectively.

Fig. 12(a)-(d) show that when the number of particle windows of MPW is upto 20000, MPW can detect the two faces in the testing image. But MPW assigns too many particle windows around object and object-like regions. The iterations of iPW are shown in Fig. 12(e)-(h) which assign proper number of particle windows around object regions. The intuition is that if there are a few acceptance particle windows around the object, then it is no longer necessary to generate additional particle windows around the object. Instead, the opportunity should be given to check other region.

Finally, experiments are conducted to see whether iPW can achieve comparable face detection results as MPW if a smaller number of particle windows is used. As in pedestrian detection experiments, SW is also used as baseline. It slides the testing image with pixel stride 2 and scale factor 1.25. As a result, SW generates 621816

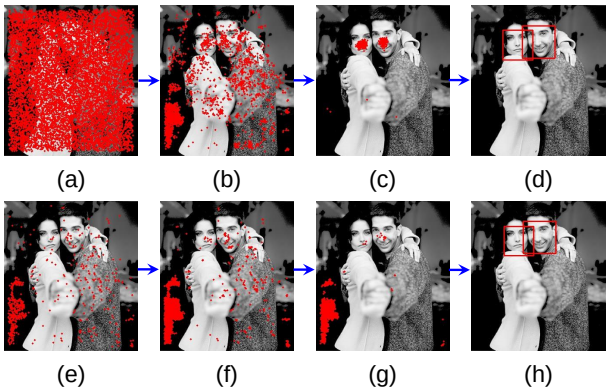


Fig. 12. (a), (b) and (c) are the updating process of particle windows in MPW. (d) is the final detection result of MPW. (e), (f) and (g) are the updating process of particle windows in iPW. (h) is the final detection result of iPW.

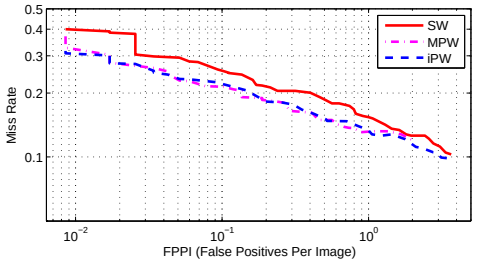


Fig. 13. DET curves comparing iPW with MPW and SW on MIT-CMU.

and 3941097 windows for 454×628 and 1024×1280 images, respectively. Limit the numbers of particle windows in MPW and iPW to $N_{MPW} = 0.25 \times N_{SW}$ and $N_{iPW} = 0.13 \times N_{SW}$, respectively. The resulting ROC curves of SW, MPW and iPW are shown in Fig. 13. It is seen from Fig. 13 that SW is almost consistently inferior to both MPW and iPW. Even only 0.13 fraction of windows are used, iPW can obtain higher detection rates than SW. Moreover, one can see Fig. 13 that the miss rates of iPW are almost identical to that of MPW.

Table 7 shows the testing time T_{SW} of SW is 1.15 times and 1.73 times of MPW and iPW, respectively, when the number of generated windows of SW, MPW and iPW are 501334, 125333 and 65173. The miss rates of the three algorithms correspond to the point in Fig. 13 with FPPI=1. The speedup effect in face detection is not as significant as pedestrian detection. The reason is that the number of HOG

TABLE VII
EFFICIENCY OF SW, MPW AND IPW ON MIT-CMU.

N_{SW}	N_{MPW}	N_{iPW}	N_{iPW}/N_{MPW}	T_{MPW}/T_{iPW}	T_{SW}/T_{MPW}	T_{SW}/T_{iPW}
501334	125333	65173	0.520	1.50	1.15	1.73

features is fixed in pedestrian detection but the number of Haar-like features varies with the response of cascade classifier.

VI. CONCLUSION

In this paper, we have proposed how to improve MPW. The proposal distribution of MPW mainly relies on the regions of support. In contrast, the proposed iPW and siPW algorithms construct the proposal distribution based on the proposed concepts of rejection, acceptance, and ambiguity particle windows which are defined by low and high thresholds of the classifier response. Both the rejection and acceptance particle windows are used for reducing the search space. The existence of the objects is reflected in the acceptance particle windows and the main clue of object locations is contained in ambiguity particle windows. Specifically, our proposal distribution is a weighted average of dented uniform distribution and dented Gaussian distribution which are dented by the rejection and acceptance particle windows. An important characteristic of the proposed algorithms is that single particle windows is generated in each stage, which makes iPW to run in an incremental manner. Experimental results have shown that iPW is about two times efficient than MPW.

REFERENCES

[1] B. Alexe, V. Petrescu, and V. Ferrari, "Exploiting Spatial Overlap to Efficiently Compute Appearance Distances between Image Windows," *Proc. Advances in Neural Information Processing Systems*, 2011.

[2] A. Andreopoulos and J. K. Tsotsos, "A Computational Learning Theory of Active Object Recognition under Uncertainty," *Int'l J. Computer Vision*, vol. 101, no. 1, pp. 95-142, 2013.

[3] Wei Bian, Tianyi Zhou, Alex M. Martinez, George Baciu, and Dacheng Tao, "Minimizing Nearest Neighbor Classification Error for Nonparametric Dimension Reduction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, vol. 8, pp. 1588-1594, Aug. 2014.

[4] R. Benenson, M. Mathias, R. Timofte, and L.V. Gool, "Pedestrian Detection at 100 Frames per Second," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

[5] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Regh, "On the Design of Cascades of Boosted Ensembles for Face Detection," *Int'l J. Computer Vision*, vol. 77, no. 1-3, pp. 65-86, 2008.

[6] [6] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten Years of Pedestrian Detection, What Have We Learned?," *Proc. European Conf. Computer Vision*, 2014.

[7] D. Benbouzid, R. Busa-Fekete, and B. Kégl, "Fast Classification using Sparse Decision DAGs," *Proc. Int'l Conf. Machine Learning*, 2012.

[8] "BING: Binarized Normed Gradients for Objectness Estimation at 300fps Ming-Ming Cheng1 Ziming Zhang2 Wen-Yan Lin3 Philip Torr1, "BING: Binarized Normed Gradients for Objectness Estimation at 300fps," *Proc. IEEE International Conference on Computer Vision and Pattern Recognition*, 2014.

[9] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," *Proc. British Machine Vision Conf.*, 2009.

[10] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast Feature Pyramids for Object Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2014.

[11] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk Cascades for Frame-Rate Pedestrian Detection," *Proc. European Conf. Computer Vision*, 2012.

[12] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.

[13] Zizhu Fan, Yong Xu, Wangmeng Zuo, Jian Yang, Jinhui Tang, Zhihui Lai, and David Zhang, "Modified Principal Component Analysis: An Integration of Multiple Similarity Subspace Models," *IEEE Trans. Neural Networks and Learning Systems,* vol. 25, no. 8, pp. 1538-1552, Aug. 2014.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *Tech. Report*, 2013.

[15] G. Gualdi, A. Prati, amd R. Cucchiara, "Multistage Particle Windows for Fast and Accurate Object Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1589-1604, 2012.

[16] S. Hinterstoisser, C. Cagniart, S. Illic, P. Sturm, N. Navab, and P. Fua, "Gradient Response Maps for Real-Time Detection of Textureless Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876-888, 2012.

[17] J. Hosang, R. Benenson, and B. Schiele, "How good are detection proposals, really?," *Proc. British Machine Vision Conference*, 2014.

[18] J. Hosang, R. Benenson, and P. Dollár, and B. Schiele, "What makes for effective detection proposals?, arXiv:1502.05082, 2015.

[19] S. Y. Kung, "From Green Computing to Big-Data Learning: A Kernel Learning Perspective," *Proc. IEEE Conf. Application-Specific Systems, Architectures and Processors*, 2013.

[20] C. Lampert, M. Blaschko, and T. Hofmann, "Efficient Subwindow Search: A Branch and Bound Framework for Object Localization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2129-2142, 2009.

[21] A. D. Lehmann, P. V. Gehler, and L.V. Gool, "Branch&Rank for Efficient Object Detection," *Int'l J. Computer Vision*, vol. 106, no. 3, pp. 252-268, 2013.

[22] A. Lehmann, B. Leibe, and L. Gool, "Fast PRISM: Branch and Bound Hough Transform for Object Class Detection," *Int'l J. Computer Vision*, vol. 94, no. 2, pp. 175-197, 2011.

[23] B. Leibe, A. Leonardis, and B. Schiele, "Robust Object Detection with Interleaved Categorization and Segmentation," *Int'l J. Computer Vision*, vol. 77, no. 1-3, pp. 259-289, 2008.

[24] Y. F. Liu, J. M. Guo, and C. H. Chang, "Low Resolution Pedestrian Detection using Light Robust Features and Hierarchical System," *Pattern Recognition,* vol. 47, pp. 1616-1625, 2014.

[25] M. W. Mak and S. Y. Kung, "Low-Power SVM Classifiers for Sound Event Classification on Mobile Devices," *Proc. IEEE Conf. Acoustics, Speech and Signal Processing*, 2012.

[26] Y. Pang, K. Zhang, Y. Yuan, and K. Wang, "Distributed Object Detection With Linear SVMs," *IEEE Transactions on Cybernetics*, vol. 44, no. 11, pp. 2122-2133, Nov. 2014.

[27] Y. Pang, Z. Song, J. Pan, and X. Li., "Truncation Error Analysis on Reconstruction of Signal from Unsymmetrical Local Average Sampling," IEEE Transactions on Cybernetics, doi:10.1109/TCYB.2014.2365513, 2015.

[28] Y. Pang, H. Yan, Y. Yuan, and K. Wang, "Robust CHOG Feature Extraction in Human Centered Image/Video Management System," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 458-468, 2012.

[29] Y. Pang, Z. Ji, P. Jing, and X. Li, "Ranking Graph Embedding for Learning to Rerank," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 8, pp. 1292-1303, Aug. 2013.

[30] S. Paisitkriangkrai, C. Shen, and A. Henge, "A Scalable Stagewise Approach to Large-Margin Multiclass Loss-Based Boosting," *IEEE Trans. Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 1002-1013, May 2014.

[31] Y. Pang, S. Wang, and Y. Yuan, "Learning Regularized LDA by

Clustering," *IEEE Transactions on Neural Networks and Learning Systems*, preprint, 2014, doi: 10.1109/TNNLS.2014.2306844.
[32] Y. Pang, Y. Yuan, X. Li, and J. Pan, "Efficient HOG Human Detection," *Signal Processing*, vol. 91, no. 8, pp. 773-781, 2011.
[33] J. Pan, Y. Pang, K. Zhang, Y. Yuan, and K. Wang, "Energy-Saving Object Detection by Efficiently Rejecting A Set of Neighboring Sub-Images," *Signal Processing*, vol. 93, no. 8, pp. 2205-2211, 2013.
[34] H. A. Rowley, S. Baluja, and T. Kanade, "Neural Network-based Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23-38, 1998.
[35] Ling Shao and Li Liu, "Feature Learning for Image Classification Via Multiobjective Genetic Programming," *IEEE Trans. Neural Networks and Learning Systems,* vol. 25, no. 7, pp. 1359 - 1371, Jul. 2014.
[36] C. Shen, P. Wang, S. Paisitkriangkrai, and A. Hengel, "Training Effective Node Classifiers for Cascade Classification," *Int'l J. Computer Vision*, vol. 103, no. 3, pp. 326-347, 2013.
[37] R. Sznitman and B. Jedynak, "Active Testing for Face Detection and Localization," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 10, pp. 1914-1920, 2010.
[38] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian Detection aided by Deep Learning Semantic Tasks," *IEEE International Conf. Computer and Vision Pattern Recognition*, 2015 .
[39] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154-171, 2013.
[40] A. Vedaldi and A. Zisserman, "Efficient Additive Kernels via Explicit Feature Maps," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 480-492, 2012.
[41] P. Viola and M. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision*, vol. 57, no. 2, pp. 137-154, 2004.
[42] X. Wang, T. Han, and S. Yan, "An HOG-LBP Human Detector with Partial Occlusion Handling," *Proc. IEEE Conf. Computer Vision*, 2009.
[43] P. Wang, C. Shen, N. Barnes, and H. Zheng, "Fast and Robust Object Detection using Asymmetric Totally Corrective Boosting," *IEEE Trans. Neural Networks and Learning Systems*, vol. 23, no.1, pp. 33-46, 2012.
[44] Pingkun Yan, Yihui Cao, Yuan Yuan, B. Turkbey, and P. L. Choyke, "Label Image Constrained Multiatlas Selection," *IEEE Transactions on Cybernetics*, vol. 45, no. 6, pp. 1158-1168, 2015.
[45] X. Yu, J. Yang, T. Wang, and T. Huang, "Key Point Detection by Max Pooling for Tracking," *IEEE Transactions on Cybernetics*, vol. 45, no. 3, pp. 444-452, 2015.
[46] W. Zhang, G. Zelinsky, and D. Samaras, "Real-Time Accurate Object Detection using Multiple Resolutions," *Proc. Int'l Conf. Computer Vision*, 2007.
[47] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan, "Fast Human Detection using A Cascade of Histograms of Oriented Gradients," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2006.
[48] C. L. Zitnick and P. Dollar, "Edge Boxes: Locating Object Proposals from Edges," *Proc. European Conference on Computer Vision*, 2014.